1. **什么是浏览器缓存**

当我们第一次访问浏览器的时候，浏览器会把网站中使用到的图片或资源缓存到磁盘或内存中，当我们再次访问的时候，网站直接从缓存中取资源，从而减轻服务器的压力

**2.expiress**

该字段是 http1.0 时的规范，它的值是一个时间戳，准确来讲是格林尼治时间，比如 Expires:Mon,18 Oct 2066 23:59:59 GMT。这个时间代表着这个资源的失效时间，在此时间之前，即命中缓存。但是这种方式会有个问题，由于失效时间是一个绝对时间，所以当服务器与客户端时间偏差较大时，就会导致缓存混乱。

**3.cache-control**

Http1.1 引入了一个新的字段，当Cache-Control和 expiress都存在时，Cache-Control优先级更高，主要取值有以下几个：

**public**：资源客户端和服务器都可以缓存。

**privite**：资源只有客户端可以缓存。

**no-cache**：需要进行协商缓存，发送请求到服务器确认是否使用缓存

**no-store**：不使用缓存。

**max-age**：缓存保质期。

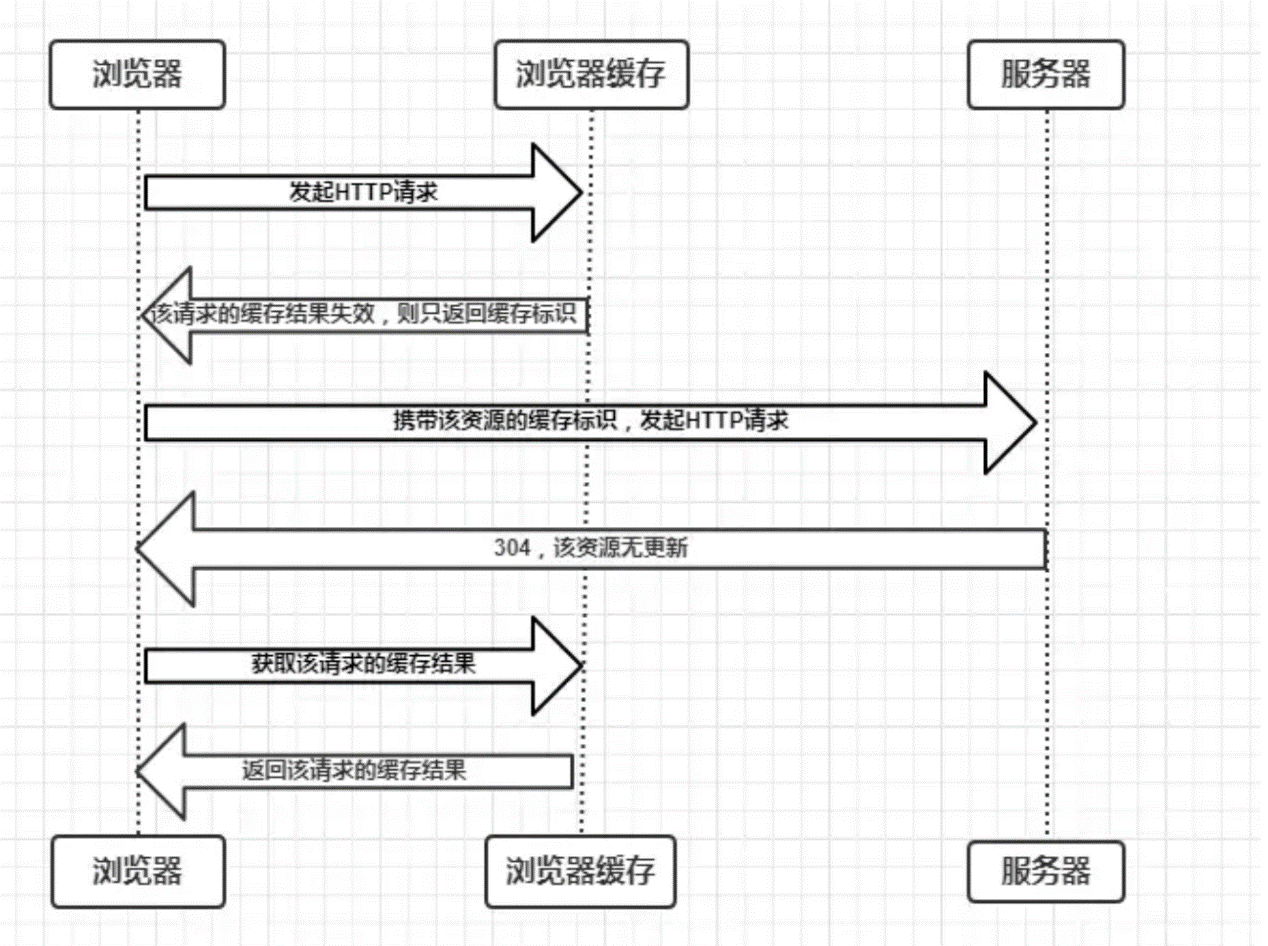
**什么是强缓存**

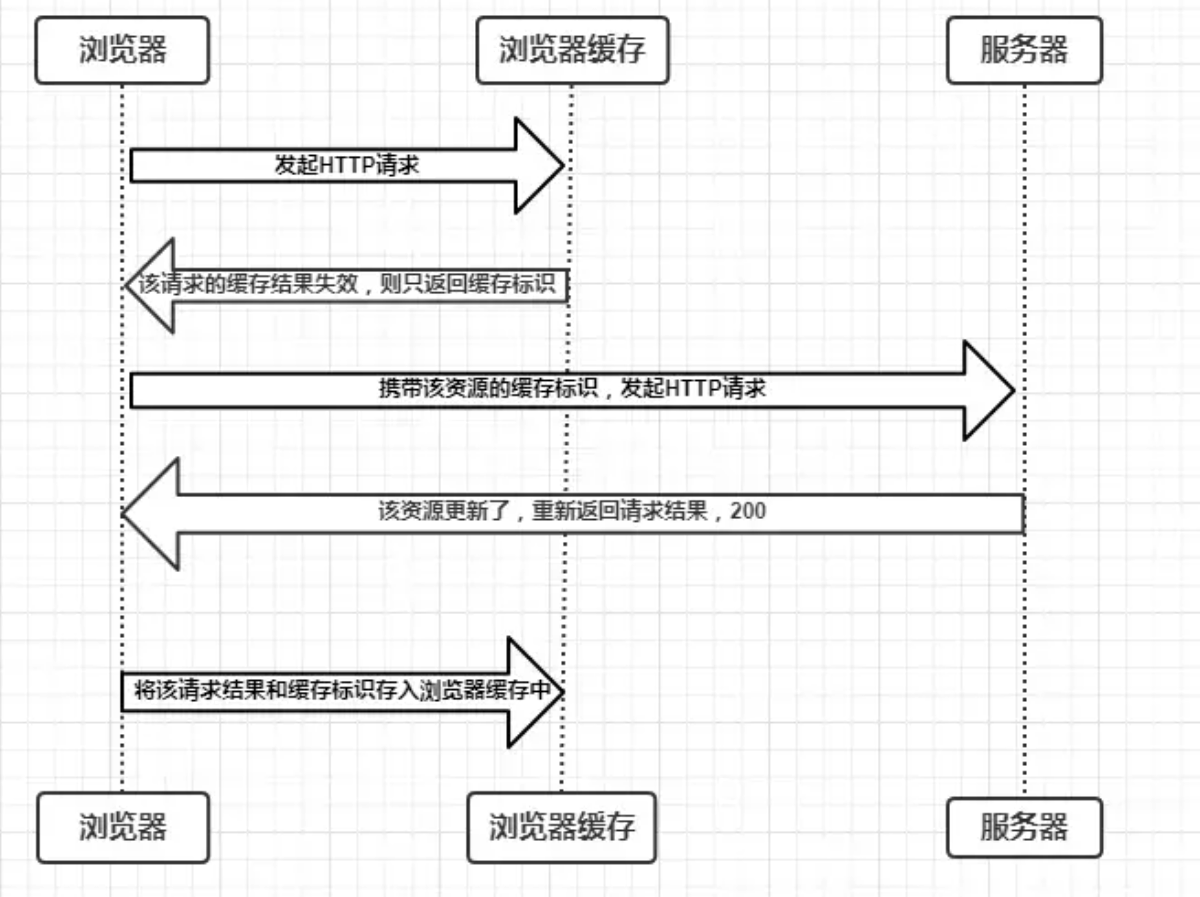
浏览器在加载资源时，会先根据本地缓存资源的 header 中的信息判断是否命中强缓存，如果命中则直接使用缓存中的资源不会再向服务器发送请求。比如刚才的expiress就是强缓存，当Cache-Control 设置值为public 以及pritive也会命中强缓存

**什么是协商缓存**

当强缓存没有命中的时候，浏览器会发送一个请求到服务器，服务器根据 header 中的部分信息来判断是否命中缓存。如果命中，则返回 304 ，告诉浏览器资源未更新，可使用本地的缓存。没有命中则返回200，会返回最新的资源，当cache-control 设置值为 no-cache时会命中协商缓存，以及下面的这几个字段

协商缓存生效，返回304



协商缓存失效，返回200和请求结果

**3. Last-Modified / If-Modified-Since**

浏览器第一次请求一个资源的时候，服务器返回的 header 中会加上 Last-Modify，Last-modify 是一个时间标识该资源的最后修改时间。

当浏览器再次请求该资源时，request 的请求头中会包含 If-Modify-Since，该值为缓存之前返回的 Last-Modify。服务器收到 If-Modify-Since 后，根据资源的最后修改时间判断是否命中缓存。

如果命中缓存，则返回 304，并且不会返回资源内容，并且不会返回 Last-Modify。

**缺点:**

因为last-Modified是毫秒级的，短时间内资源发生了改变，Last-Modified 并不会发生变化。

**4. Etag / If-None-Match**

与 Last-Modify/If-Modify-Since 不同的是，Etag/If-None-Match 返回的是一个校验码。ETag 可以保证每一个资源是唯一的，资源变化都会导致 ETag 变化。服务器根据浏览器上送的 If-None-Match 值来判断是否命中缓存。

与 Last-Modified 不一样的是，当服务器返回 304 Not Modified 的响应时，由于 ETag 重新生成过，response header 中还会把这个 ETag 返回，即使这个 ETag 跟之前的没有变化。

Last-Modified 与 ETag 是可以一起使用的，服务器会优先验证 ETag，一致的情况下，才会继续比对 Last-Modified，最后才决定是否返回 304。

**案例**

1. 通过node启动一个服务，设置Cache-control: max-age=86400，表示设置缓存时2个小时，看下效果

>1 首次进入,全部走网络请求

>2 f5刷新，全部走内存，因为标签页未关闭所有内存还存在

>3 关闭标签页，全部走磁盘，因为标签页关闭内存已经清除了，只能从磁盘拿数据

>4 crt+f5强制刷新，全部走网络请求

1. 我们把每个资源请求都设置两次，设置Cache-Control: no-store，看下效果
2. 每个资源请求设置2次，设置Cache-Control: no-cache，再看下效果